Amendments to the Specification

Please replace the title with the following new title:

BUSINESS SOFTWARE APPLICATION FRAMEWORK FOR DEVELOPING AND IMPLEMENTING A COMPOSITE APPLICATION

The amendments below are made with reference to the paragraph numbers of the application as published.

Please replace paragraph [0035] with the following amended paragraph:

[0035] The systems and techniques described here relate to a framework for developing and implementing applications in an enterprise management system. For example, a framework may be used to develop and implement a composite application, which overlays an enterprise information technology (IT) ~~IT~~ platform and uses it to implement processes that are not the core enterprise transactional processes. That is, a composite application may orchestrate a business process in synchronization with existing processes (e.g., native processes of enterprise base systems) and leverage existing investments in the IT platform. Furthermore, composite applications may be run on a heterogeneous IT platform. In doing so, composite applications may be cross-functional. That is, they may drive business processes across different applications, technologies, and organizations. Accordingly, composite applications may drive end-to-end business processes across heterogeneous systems. Additionally, composite applications may be combined with each other in order to enlarge the process coverage. Composite applications may also support semi-structured processes, tackle event-driven and knowledge-based scenarios, and support a high degree of collaboration in teams. In teams, for example, people may work on specific tasks in specific roles in

specific teams. Composite applications may relate knowledge, structured information, and/or unstructured information within the context of a business process and may be triggered by events, aggregate and contextualize information, and drive collaboration and transactions. Different applications supported by different frameworks may have any combination of these characteristics. Thus, different implementations of the framework may be used for developing and implementing various types of applications.

Please replace paragraph [0038] with the following amended paragraph:

[0038] In particular implementations, the portal may ~~includes~~ include a security component, a content directory component, a view builder, a content management component, and one or more service interfaces to an enterprise management consolidation system 140. The security component may protect data transmission using encryption (e.g., Secure Socket Layers (SSL)), digital signatures, and/or watermarking. The view builder may create role-based interactive views (e.g., Web pages) for presentation to users. The content management component may include a retrieval and classification component (e.g., Text Retrieval and Extraction (TREX) component) and a collaboration component. The retrieval and classification component may automatically analyze unstructured components to identify know-how. The service interfaces could include an Internet Transaction Server (ITS) component, various connectors, such as a Java® programming language Connector, and a Business Intelligence platform.

Please replace paragraph [0056] with the following amended paragraph:

[0056] As discussed previously, composite applications typically implement new or additional processes, as opposed to the core transactional processes, in an existing IT landscape. Composite applications may also support semi-structured processes, tackle event-driven and knowledge-based business scenarios, and support collaboration in teams. In particular implementations, composite applications may support the Java® runtime stack.

Please replace paragraph [0059] with the following amended paragraph:

[0059] UI layer 230 provides user interfaces that allow a user to interact with composite applications. In particular implementations, UI layer 230 provides pattern components, such as, for example, a dashboard, a search bar, a browse and collect function, an object editor, and phases for a guided procedure, as building blocks for user interfaces. UI layer 230 may also decouple application logic from the UI. As shown, UI layer 230 accomplishes this by having a separation of the business objects, which are in the object access layer 210, and application services, which are in service layer 220, from the user interface elements, which are in UI layer 230. This allows UI components to be reused in different application contexts. This also allows business objects and application services to be visualized differently according to the specific equipments of a certain use case. UI layer 230 may also leverage the metadata information on business objects and services through metadata-driven UI-generation and configuration. The metadata approach allows for ready adaptability to alternative screens depending on the end users needs (e.g., in different industries). UI layer 230 may additionally allow integration (e.g., binding) into OAL 210 to access business objects, business services, and metadata. Thus, UI

components may be connected to business objects in OAL 210. UI layer 230 may support any appropriate type of user interfaces, such as, for example, a user interface composed of pattern-based components and/or freestyle components with interfaces to the user interface components--this user interface will discussed in more detail below--or Java Server Pages (JSP®) from Sun Microsystems®. ~~Java Server Pages (JSPs) from Sun~~.

Please replace paragraph [0063] with the following amended paragraph:

[0063] Framework 200 may be implemented using readily available technology. For example, the computer-implemented framework may be implemented using ~~mySAP~~ mySAP™ technology components. In particular implementations, the components may include an SAP® Web Application Sever (WAS) to run the applications, an SAP® Enterprise Portal to render the applications, an ~~SAP KW™~~ SAP® KW to handle unstructured information sources, pattern-based components and/or freestyle components with interfaces to the UI components to design UIs and to provide J2EE® framework and ABAP® framework run-time integration, an SAP® BW to provide reporting and analytics, data mining, and planning and simulation, SAP® Business Process Management (BPM), an SAP® Exchange Infrastructure (XI) to provide shared integration knowledge separate from applications, and SAP® Web services to offer business functionality over the Internet.

Please replace paragraph [0065] with the following amended paragraph:

[0065] Examples of the types of business processes supported by the framework ~~including~~ include enterprise change management, product innovation, employee productivity, and enterprise service automation. Enterprise change management may support enterprises when merging, splitting, acquiring, spinning off, or reorganizing. Product innovation may support the life cycle of a product, including the prenatal phase of collecting ideas and consolidating them into concepts, the market launch phase, and the end of life. In doing so, the resources of a PLM and CRM may be drawn upon. Employee productivity aims to increase employee productivity, decrease costs, and increase employee satisfaction. Key functions may include manager self services, employee self services, expert finders, e-procurement, and e-learning. ~~ERM~~ Enterprise Risk Management (ERM) and B2E® resources may be drawn upon to accomplish these tasks. Enterprise service automation provides administration and monitoring functions as well as evaluation tools to facilitate project success. An example of this is the setting up of projects and the staffing with people with the required skills and availability. Additional application families may also be created.

Please replace paragraph [0070] with the following amended paragraph:

[0070] Generators 314 are used for generating actual code from the portions modeled by modeling tools 312. To accomplish this, the generators may use templates that are stored in metadata repository 360. Driven by the metadata in repository 360, the generators may automatically create Java® language classes (e.g., for use in run-time components 320) and also configuration files (e.g., to adjust UI patterns to a certain business object). Thus, the connectivity to back-end systems and the application persistency may be generated, as well as a default user interface. The generators may also generate interfaces for application services, data access logic, and persistency.

Please replace paragraph [0074] with the following amended paragraph:

[0074] In certain implementations, layer 330 includes extensions to document management or content management that allow business objects to use the functionality for documents. For example, taxonomies for business objects, transparent indexing of TREX for structured and unstructured objects, and subscription services for dependent objects independent of the repository where the objects reside may be provided. Layer 330 may also provide transaction support, in as far as the transaction concept is also supported by concerned source systems, a metadata interface, allowing an application to be dynamically configured at run-time, and subscription services (e.g., J2EE® publish and subscribe service).

Please replace paragraph [0086] with the following amended paragraph:

[0086] Application services container 348 is used to implement model specific services for one or more business applications. Although generic objects, generic services, and/or processes may be generated for an application, some business logic is too specific to be implemented generically. For example, determining the number of vacation days that an employee has may involve determining the number of vacation days the employee is entitled to per year, determining the number of days available based on the employee's service to date for the year, determining how many days the employee has been absent to date for the year, and determining whether to assign those days to vacation days or sick days. Furthermore, if the employee is splitting time between departments, an allocation may need to be made between the two. As another example, an order process at a

manufacturer may include obtaining ~~a~~ an order, splitting the order into components based on the bill of materials, determining whether each component is in stock, if a component is not in stock, determining where and/or how to procure it, and, if a component must be procured, determining a potential delivery date. The business logic for such tasks may be difficult to model generically, especially across a wide variety of industries. Thus, the logic may have to be individually coded. Container 348 provides interfaces for the code to be used. The interfaces may be generated by the metadata of the service model, but the inner code has to be programmed individually. Also, maintaining the service definition in the design-time allows generation of an empty service.

Please replace paragraph [0090] with the following amended paragraph:

[0090] UI framework 352 may support any appropriate type of user interfaces. For example, the UI framework may support a user interface composed of pattern-based components and/or freestyle components with interfaces to the user interface components ~ this user interface will be discussed in more detail below ~ or ~~Java Server Pages (JSPs)~~ JSP® server pages from Sun Microsystems®. UI framework 352 may also support a Java® langauge front-end and ABAP® langauge back-end, a Java® langauge front-end and Java® language back-end, or any other appropriate combination of front-end and back-end. The framework may additionally provide a construction kit for complex components and applications and configuration of patterns via XML, URL, UML, or other appropriate technique.

Please replace paragraph [0095] with the following amended paragraph:

[0095] In particular implementations, the composite application portions may be implemented as ~~Enterprise Java Beans (EJBs)~~ Enterprise Javabeans® modules. ~~hi~~ In other implementations, the design-time components may have the ability to generate the run-time implementation into different platforms, such as J2EE®, ABAP®, or .NET®.

Please replace paragraph [0097] with the following amended paragraph:

[0097] In particular implementations, generators 314 allow template-based generation of ~~Java coding~~ Java® code, database tables, entries in metadata repository 360, XML configuration files, etc. This may be implemented with extensibility and the ability to conduct upgrades without loosing his information. This capability may be achieved by allowing the metadata of the new implementation to be compared with the metadata of the existing implementation during an upgrade. If there are implementation-specific extensions, they may be identified, and strategies for solution of possible conflicts maybe proposed.

Please replace paragraph [0099] with the following amended paragraph:

[0099] Framework 300 may be implemented using readily available technology. For example, the framework may be implemented using ~~mySAP~~ mySAP™ technology components. In particular implementations, the components may include an SAP® Web Application Sever (WAS) to run the applications, an SAP® Enterprise Portal to render the applications, an SAP® KW to handle unstructured information sources, an SAP® BW to provide reporting and analytics, data mining, and planning and simulation, SAP® Business Process Management

(BPM), an SAP® Exchange Infrastructure (XI) to provide shared integration knowledge separate from applications, and SAP® Web services to offer business functionality over the Internet.

Please replace paragraph [0100] with the following amended paragraph:

[00100] For instance, an SAP® WAS may include a J2EE® engine, SAP® IDE, Universal Workflow, and Deployment Service. The WAS may also include a pattern-based and freestyle-based user interface development and interface module. Also, an SAP® Enterprise Portal may provide unified access to applications, information, and services by using views, roles, pages, worksets, top-level navigation, and KM. This enterprise portal also provides login management and user management. For KM, unstructured information consists of collaboration and content management. For collaboration, KM enables team-driven business processes, synchronous and asynchronous applications, groupware integration, calendars, bulletin boards, threaded discussions, and collaboration rooms. For content management, KM handles documents, feedback, rating, publishing, subscription, document workflow, versioning, archiving, indexing, searching, and taxonomies. An SAP® BPM may cover life cycles (e.g., design, development, deployment, and change). An SAP® XI may provide external and internal integration of system and connectors to various systems such as Oracle®, Siebel®, Peoplesoft®, and SAP® systems. The SAP® XI may be based on Web services, ~~JAVA~~ Java®, and XML standards. SAP® Web services may provide a service provider, service handler, and service user. Additionally, an SAP® BW may be used.

Please replace paragraph [0101] with the following amended paragraph:

[00101] Moreover, the KM and collaboration functionality may embedded in applications, not only in separate pages in the portal. Furthermore, any general development environment may be used. For example, the development environment could include the Java® programming language, ~~with~~ EJB® 2.0 components, the JDO™ API, Java® persistency, and Java® application logic, ~~Advanced Business Application Programming (ABAP)~~ ABAP® components, and Web services. Existing ABAP® components may be integrated via Java® connector calls. In particular implementations, the complete Javau® stack could be used. Furthermore, Web service technology may be used for remote access.

Please replace paragraph [0104] with the following amended paragraph:

[00104] Examples of the types of business processes supported by the framework including enterprise change management, product innovation, employee productivity, and enterprise service automation. Enterprise change management may support enterprises when merging, splitting, acquiring, spinning off, or reorganizing. Product innovation may support the life cycle of a product, including the prenatal phase of collecting ideas and consolidating them into concepts, the market launch phase, and the end of ~~life, hi~~ life. In doing so, the resources of a PLM and CRM may be drawn upon. Employee productivity aims to increase employee productivity, decrease costs, and increase employee satisfaction. Key functions may include manager self services, employee self services, expert finders, e-procurement, and e-learning. ERM and B2E® resources may be drawn upon to accomplish these tasks. Enterprise service automation provides administration and monitoring functions as well as evaluation tools to facilitate project

success. An example of this is the setting up of projects and the staffing with people with the required skills and availability. Additional application families may also be created.

Please replace paragraph [0108] with the following amended paragraph:

[00108] Business object modeler 410 includes an Integrated Development Environment (IDE) application program interface (API) 411, an object modeler 412, and a relation modeler 413. IDE API 412 allows modeler 412 to be integrated into an Eclipse IDE, which supports the modeling of the business object by object modeler. 412. For example, the integration supports generation of business objects as ~~EJBs~~ EJB® objects, interfaces for application services, default user interfaces, data access logic, and persistency. Relation modeler 413 allows the modeling of relations between modeled objects. For example, a sales order could be composed of a customer, a product, and a price. Relation modeler 413, therefore, allows for the modeling of the relations between these items. In operation, for instance, if a user interface is generated for a sales order, the semantics for each field in the sales order may be identified. Additionally, a connection to the value help function may be facilitated.

Please replace paragraph [0110] with the following amended paragraph:

[00110] Generator 430 includes a generator framework 432, a persistency generator 434, an EJB® 436 generator, a UI adapter generator 438, a Web service generator 440, and a metadata API 442. Generator framework 432 may also be integrated into the Eclipse DDE.

Please replace paragraph [0111] with the following amended paragraph:

[00111] To generate a business object, generator 430 may use templates in metadata repository 450 and code them with object metadata and relation metadata in the repository. Generator 430 may also generate the data persistency for the business object, and generate the actual business object, an EJB® object in this instance. Generator 430 may additionally generate user interfaces for the business object and any necessary Web services.

Please replace paragraph [0112] with the following amended paragraph:

[00112] The templates may be generic. In particular implementations, the generators automatically create Java® classes (e.g., for the implementation of the object access layer), JDO™ tables, ~~EJBs~~ EJB® objects, and configuration files, to adjust UI patterns to a certain business object, for example. Thus, the connectivity to back-end systems and the composite application persistency is generated as well as a default User Interface. Furthermore, UI adapters for a UI development and interface module and, if necessary, Web services may be generated. The output of such a process may be real working code in the object access layer of the run-time components.

Please replace paragraph [0113] with the following amended paragraph:

[00113] One example is the generation of a run-time implementation of a business object in an object access layer. The generator reads the business object metadata from the repository and

generates the JDO™ persistency, the connectivity to the XI, the KW and/or the BW (e.g., by using proxies), the generic methods, and the basic UI. For this coding, templates (e.g., for services) or XML-templates (e.g., for JDO™ persistency) are used where business object specific coding or XML is added, and the result is stored as complete code or complete XML.

Please replace paragraph [0117] with the following amended paragraph:

[00117] As illustrated, user interface 500a includes a directory section 510 and a work section 520. Directory section 510 includes a listing of business objects for product definition. The creation of a new business object may be initiated by right-clicking in directory section 510, as shown, and selecting the appropriate line in the generated menu. In response to the initiating command, work section 520 includes an association section 522 in which a name is associated with the new business object and the business object is associated with ~~an~~ a composition application, "xApp-xPD" in this instance.

Please replace paragraph [0128] with the following amended paragraph:

[00128] FIG. 5G illustrates user interface 500f. Once aspects for business objects have been specified, user interface 500f allows the specification of technical data for tables if the business object is to have its own local persistency. In this example, the tables are JDO™ tables, although they could be of any other appropriate type.

Please replace paragraph [0135] with the following amended paragraph:

[00135] The illustrated components 600 are for an object access layer 610. Layer 610 has a variety of business objects 620 executing therein. Each business object 620 includes a business object bean 622 that includes lifecycle methods, properties, and validation. The beans 622 may, for example, be ~~Enterprise Java Beans (2.0)~~ Enterprise Javabeans® beans and may be generated by the metadata, as already mentioned. Each business object 620 also includes a data object 624. Data object 624 allows bean 612 to be decoupled from the underlying data, which is accessed through data a access service factory 630.

Please replace paragraph [0136] with the following amended paragraph:

[00136] Data access service factory 630 is responsible for homogenizing data from various sources. In accomplishing this, factory 630 determines data sources for business objects 620. Thus, it may be viewed as a kind of dispatcher. For example, data access service factory 630 may read data from a remote service 642, a persistence service 644, a KM service 646, and/or an On line analytical processing (OLAP) ~~OLAP~~ service 648 and present this data to data object 624. The data access service factory 630 may accomplish this by allowing the plug in of special adapters for accessing different sources of data. The composite application framework may provide adapters for the exchange infrastructure (XI), the knowledge management system (KM), and the data warehouse (BW). It may also provide access to a local database that is treated the same way as remote access adapters.

Please replace paragraph [0171] with the following amended paragraph:

[00171] As illustrated, framework 1000 includes a business object service module 1010, a pattern configuration module 1020, a metadata adapter 1030, and an EJB® adapter 1040. In general, service module 1010, metadata adapter 1030, and EJB® adapter 1040 are responsible for translating data into a format usable by pattern configuration module 1020. This makes a separation between the components of framework 1000 and an object access layer 1080, an example of which was discussed previously. Service module 1010 may be an API that allows business object data to be brought into the UI framework environment. Service module 1010 includes a generic client proxy 1012, which prepares data for other components of the UI framework 1000 and/or for external UI components. Proxy 1012 may transform data to a generic API. In effect, then, it may wrap an API to provide an API for the upper-level components. Metadata adapter 1030 allows the metadata of a business object model to be exposed to the UI world, so that a unified metadata repository exists along the whole stack. EJB® adapter 1040 allows EJB® components, such as tables, business objects, and data, to be translated into a format for framework 1000. Pattern configuration module 1020 contains the configured patterns generated using design-time components.

Please replace paragraph [0173] with the following amended paragraph:

[00173] The technology of module 1090 may be based on ~~JavaServer Pages (JSPs)~~ JSP® pages and a tag library containing, among other things, ready-made user interface elements (such as push buttons and input fields). The description of the application in a metadata repository is used to generate the run-time code for one of the following run-time environments: ~~Java, ABAP,~~

~~or .NET.~~ the Java® run-time environment, the ABAP® run-time environment, or the .NET® run-time environment.

Please replace paragraph [0178] with the following amended paragraph:

[00178] UI module tools that enable a developer to create such applications are part of the integrated development environment (IDE) of the SAP® Web Application Server. The design tools consist of a view designer, editors for controllers, an application modeler to define the flow of an application, and a model designer.

Please replace paragraph [0179] with the following amended paragraph:

[00179] UI module may also support location-independent rendering. That is, depending on the capabilities of the client, the module run-time will generate the HTML for the browser either on the client or on the server. Client-side rendering, using ~~the JavaScript-based~~ a client-side framework based on a JavaScript® engine will be possible for the Microsoft Internet Explorer® browser as of version 5.5 (or higher) and for higher versions of the Netscape Navigator® browser.

Please replace paragraph [0180] with the following amended paragraph:

[0180] Earlier versions of the Microsoft Internet Explorer® or Netscape Navigator® browsers, as well as a variety of mobile devices, will be supplied with output that is rendered on

the server. Server-side rendering is also the only choice for browsers where the JavaScript® scripting engine has been disabled for security reasons.

Please replace paragraph [0186] with the following amended paragraph:

[00186] As discussed, a composite application framework may extend an underlying enterprise platform. For example, a composite application framework may extend ~~my SAP~~ mySAP™ Technology by adding modeling and configuration tools (e.g., business object modeler and guided procedures). As another example, generic components (e.g., UI patterns and generic services) may be included. As a further example, a standardized interface (e.g., object access layer) may be used to access the enterprise base systems. As an additional example, a framework may provide reusable content (e.g., predefined object models and XI content), and integration infrastructure (e.g., to connect business objects and documents and to provide access to XI proxies). Furthermore a composite application framework may provide guidelines (e.g., composite application cookbook). The guidelines may be designed to work cooperatively with guided procedures.